iOS WidgetKit

Building Widgets with SwiftUI

By James Shaw

iOS 15 WidgetKit: Build Widgets with SwiftUI

Version 1.0 - October 2021

Chapter 1: Introduction

👋 Welcome and About Me

Hello and a very warm welcome to ***iOS 15 WidgetKit: Build Widgets with SwiftUI***. 😁

I'm **James Shaw**, a Software Engineer specialising in iOS Development using Swift, SwiftUI and Objective-C. I published my first App to the iOS App Store way back in 2009 and since then have worked on 50+ iOS projects. The latter of which have all included iOS widgets.

In this book, I'm going to share all that I have learnt during my experiences using Apple's ***WidgetKit*** framework to help you build awesome widgets for your iOS apps. You'll learn all about configuring your project to support widgets, feeding them with timely data, and sharing information between your app and widget.

Let's go! 🚀

📱 Introduction to iOS Widgets

Apple first introduced widgets in **iOS 14 at WWDC 2020**. They allow users to view 3rd party app content on their home screen. This provides an awesome opportunity for developers to place key functionality of their app front and center on the user's Home Screen. A great way to increase user engagement and retention.

Like with many Apple frameworks; *WidgetKit* does have a number of limitations;

- Widgets are limited to **four sizes**; small, medium, large and extra large (iPadOS only).
- Users are not able to **interact** with or **scroll** widgets.
- When tapped, widgets will **always** open the parent app.
- The UI code of a widget must be built using ***SwiftUI*, not *UIKit***. (Note that the parent app can still be built using UIKit).

That being said, Widgets have been one of the most popular additions to iOS in recent years with many widely-used iOS apps adding them to their offering. The widget market is booming and it's certainly a great time to add them to your apps.

📘 Course Content

This book is going to take you through the full spectrum of functionality that widgets offer; you'll learn everything there is to know about them. To keep the content *WidgetKit* specific; we won't be delving too deeply into the standard *SwiftUI* side of things. We'll be keeping the UIs really simple and purely focussing on the quirks of using *WidgetKit*.

In Chapter 2, we'll start by learning the basics and fundamentals of *WidgetKit* and explore the terminology used by Apple. We'll configure a simple iOS app project that contains **multiple, static widgets** along with the main iOS app.

Following on, in Chapter 3 we'll use these fundamental principles to build a simple **Battery Status** widget similar to the default app found in iOS 15. You'll learn how to populate the widget with real battery level data and display it in different UIs depending on the widget size.

In Chapter 4, we'll take things further by building a **ToDo List** app and widget where the data displayed on the widget is shared between the two. You'll also learn how to manually refresh widgets from the parent app when data is changed.

Lastly in Chapter 5, we'll build a **Weather Forecast** widget with real, localised weather data fed directly from a weather API. We'll also explore how we can give the user the ability to customise and configure their widget by allowing them to choose a location for the weather forecast directly from the widget.

🙋 iOS Apps I've built with WidgetKit

As I alluded to above, I have had significant experience building widgets for my own apps on the App Store. Here are four of them if you wish to check them out.

Twidget - Widget for Twitter
View your Twitter timeline right on your iOS Home Screen.
App Store | Website

Stats - Health, Fitness Widget
View your Apple Health and Fitness data right on your iOS Home Screen.
App Store | Website

YouWidget - Widget for YouTube
View your YouTube video feed right on your iOS Home Screen.
App Store | Website

Teams - Football/Soccer Widget
View football teams, scores, results, standings and fixtures right on your iOS Home Screen.
App Store | Website